RETICULATION-VISIBLE NETWORKS

MAGNUS BORDEWICH AND CHARLES SEMPLE

ABSTRACT. Let X be a finite set, $\mathcal{N}$ be a reticulation-visible network on X, and $\mathcal{T}$ be a rooted binary phylogenetic tree. We show that there is a polynomial-time algorithm for deciding whether or not $\mathcal{N}$ displays $\mathcal{T}$. Furthermore, for all $|X| \geq 1$, we show that $\mathcal{N}$ has at most $8|X| - 7$ vertices in total and at most $3|X| - 3$ reticulation vertices, and that these upper bounds are sharp.

1. INTRODUCTION

Phylogenetic networks have become increasingly more prominent in the literature as they correctly allow the evolution of certain collections of present-day species to be described with reticulation (non-tree-like) events. However, the evolution of a particular gene can generally be described without reticulation events. As a result, analysing the tree-like information in a phylogenetic network has become a common task. Central to this task is that of deciding if a given phylogenetic network $\mathcal{N}$ infers a given rooted binary phylogenetic tree on the same collection of taxa. In this paper, we show that if $\mathcal{N}$ is a so-called reticulation-visible network, then there is a polynomial-time algorithm for making this decision. This resolves a problem left open in [2] and [4]. In the rest of the introduction, we formally state this result as well as the other main result which concerns the number of vertices in a reticulation-visible network.

Throughout the paper, X denotes a non-empty finite set. A *phylogenetic network on* X is a rooted acyclic digraph with no parallel arcs and the following properties:

(i) the root has out-degree two;

Date: April 13, 2016.

1991 *Mathematics Subject Classification.* 05C85, 68R10.

Key words and phrases. Phylogenetic network, reticulation-visible network, TREE CONTAINMENT problem.

Part of this work was conducted while the first author was an Erskine Visiting Fellow at the University of Canterbury, New Zealand. The second author was supported by the Allan Wilson Centre, and the New Zealand Marsden Fund.

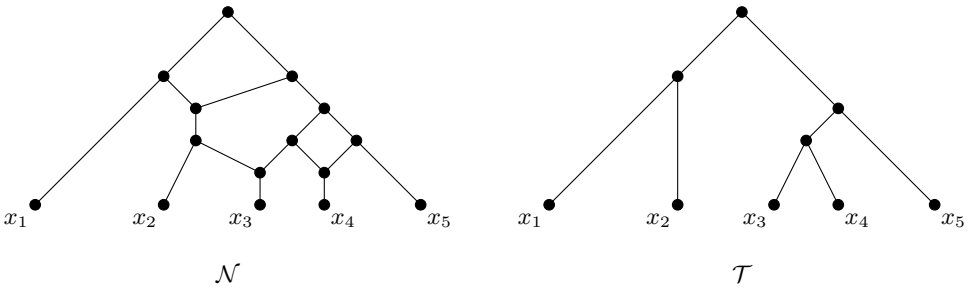


FIGURE 1. The phylogenetic network $\mathcal{N}$ displays the rooted binary phylogenetic tree $\mathcal{T}$.

(ii) vertices of out-degree zero have in-degree one, and the set of vertices with out-degree zero is X; and
(iii) all other vertices either have in-degree one and out-degree two, or in-degree two and out-degree one.

For technical reasons, if $|X| = 1$, then we additionally allow $\mathcal{N}$ to consist of the single vertex in X. The vertices in $\mathcal{N}$ of out-degree zero are called *leaves*. Furthermore, the vertices of $\mathcal{N}$ with in-degree two and out-degree one are called *reticulations*, while the vertices of in-degree one and out-degree two are called *tree vertices*. The arcs directed into a reticulation are *reticulation arcs*; all other arcs are called *tree arcs*. Note that, what we have called a phylogenetic network is sometimes referred to as a *binary* phylogenetic network. A *rooted binary phylogenetic X-tree* is a phylogenetic network on X with no reticulations.

Let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. We say that $\mathcal{N}$ *displays* $\mathcal{T}$ if $\mathcal{T}$ can be obtained from $\mathcal{N}$ by deleting arcs and vertices, and contracting degree-two vertices. To illustrate, in Fig. 1, the phylogenetic network $\mathcal{N}$ on $X = \{x_1, x_2, x_3, x_4, x_5\}$ displays the rooted binary phylogenetic X-tree $\mathcal{T}$. The particular problem of interest is the following:

TREE CONTAINMENT
Instance: A phylogenetic network $\mathcal{N}$ on X and a rooted binary phylogenetic X-tree $\mathcal{T}$.
Question: Does $\mathcal{N}$ display $\mathcal{T}$?

In general, TREE CONTAINMENT is NP-complete [5] even when the instance is highly constrained [4].

Let $\mathcal{N}$ be a phylogenetic network on X. A vertex v in $\mathcal{N}$ is *visible* if there is a leaf ℓ in X with the property that every directed path from the root to ℓ traverses v, in which case, we say ℓ *verifies the visibility of* v (or, more briefly,

ℓ verifies v). If every reticulation in $\mathcal{N}$ is visible, then $\mathcal{N}$ is a *reticulation-visible network*. In Fig. 1, $\mathcal{N}$ is a reticulation-visible network. For example, the visibility of the leftmost reticulation is verified by x_2. Observe that this reticulation is not verified by x_3 as there is a path from the root of $\mathcal{N}$ to x_3 that avoids it. For the reader familiar with tree-child networks, $\mathcal{N}$ is tree-child network if and only if every vertex in $\mathcal{N}$ is visible [1, Lemma 2]. Thus tree-child networks are a proper subclass of reticulation-visible networks. It was shown in [4] that there exists a polynomial-time algorithm for Tree Containment if $\mathcal{N}$ is a tree-child network. Here we generalise that result to reticulation-visible networks, thereby resolving a problem left open in [2] and [4]. The next theorem is the first main result of the paper.

Theorem 1.1. *Let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. Then* Tree Containment *for $\mathcal{N}$ and $\mathcal{T}$ can be decided in polynomial time.*

It is shown in [2] that a reticulation-visible network on X has at most $4(|X| - 1)$ reticulations. The second of the two main results sharpens this result.

Theorem 1.2. *Let $\mathcal{N}$ be a reticulation-visible network $\mathcal{N}$ on X and let $m = |X|$. Then $\mathcal{N}$ has at most $8m - 7$ vertices in total and at most $3m - 3$ reticulations. Moreover, these bounds are sharp for all integers $m \geq 1$.*

The paper is organised as follows. The next section consists of concepts that will be used in the description of the algorithm that establishes Theorem 1.1. Called TreeDetection, the description of this algorithm and the proof of its correctness is given in Section 3. In Section 4, we analyse the algorithm's running time and show that TreeDetection completes in $O(|X|^3)$ steps. Together Sections 3 and 4 constitute the proof of Theorem 1.1. The last section, Section 5, contains the proof of Theorem 1.2. Throughout the paper, notation and terminology follows Semple and Steel [6].

Lastly we remark that it has been brought to our attention that a result similar to Theorem 1.1 has been independently obtained in [3].

2. Preliminaries

Let $\mathcal{N}$ be a phylogenetic network, and let u and v be vertices in $\mathcal{N}$. If u and v are joined by an arc (u, v), we say u is a *parent* of v and, conversely, v is a *child* of u. More generally, if u and v are joined by a directed path from u to v, we say u is an *ancestor* of v and, conversely, v is a *descendant* of u. Furthermore, if u is neither an ancestor nor descendant of v, we say u and v are *non-comparable*. A *backward path* in $\mathcal{N}$ from v to u is an underlying

path $v = w_0, w_1, w_2, \ldots, w_{k-1}, w_k = u$ such that, for all $i \in \{1, 2, \ldots, k\}$, we have that $\mathcal{N}$ contains the arc (w_i, w_{i-1}). Observe that if there is a backward path in $\mathcal{N}$ from v to u, then u is an ancestor of v. An *up-down path* P in $\mathcal{N}$ from u to v is an underlying path $u = w_0, w_1, w_2, \ldots, w_{k-1}, w_k = v$ such that, for some $i \leq k-1$, we have that $u = w_0, w_1, w_2, \ldots, w_i$ is a backward path from u to w_i and $w_i, w_{i+1}, \ldots, w_k = v$ is a (directed) path from w_i to v. The vertex w_i is the *peak* of P. Also, a *tree path* in $\mathcal{N}$ from u to v is a (directed) path such that, except possibly u, every vertex on the path is either a tree vertex or a leaf. Lastly, the *length* of a (directed) path, a backward path, an up-down path, and a tree path is the number of edges in the path.

Let $\mathcal{N}$ be a phylogenetic network on X. A 2-element subset $\{x, y\}$ of X is a *cherry* in $\mathcal{N}$ if there is an up-down path of length two between x and y. Equivalently, $\{x, y\}$ is a cherry if the parent of x and the parent of y are the same. For a cherry $\{x, y\}$ in $\mathcal{N}$, let $\mathcal{N}'$ be the phylogenetic network obtained from $\mathcal{N}$ by deleting x and y, and their incident arcs, and labelling their common parent (now itself a leaf) with an element not in X. We say that $\mathcal{N}'$ has been obtained from $\mathcal{N}$ by *reducing the cherry* $\{x, y\}$.

Let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. Suppose that $\mathcal{N}$ displays $\mathcal{T}$. Then there is a subgraph $\mathcal{T}'$ of $\mathcal{N}$ that is a subdivision of $\mathcal{T}$. We say $\mathcal{T}'$ is an *embedding* of $\mathcal{T}$ in $\mathcal{N}$. Observe that any embedding of $\mathcal{T}$ in $\mathcal{N}$ can be formed by deleting exactly one incoming arc at each reticulation and deleting any resulting degree-one vertex that is not a leaf of $\mathcal{N}$. We refer to the action of deleting one of the two incoming arcs at a reticulation as *resolving the reticulation*. More precisely, if v is a reticulation with parents p and q and we delete (p, v), then we have *resolved v towards q*.

Let $\mathcal{N}$ be a reticulation-visible network and let x be a leaf of $\mathcal{N}$ with parent p. If p is a tree vertex, then $\mathcal{N}\backslash x$ denotes the phylogenetic network obtained from $\mathcal{N}$ by deleting x and its incident edge, and contracting the resulting degree-two vertex. If p is a reticulation, then, as no parent of a reticulation in $\mathcal{N}$ is also a reticulation, $\mathcal{N}\backslash x$ denotes the phylogenetic network obtained from $\mathcal{N}$ by deleting x and p, and their incident edges, and contracting the resulting degree-two vertices. Observe that, in both cases, $\mathcal{N}\backslash x$ is a reticulation-visible network.

We end this section with a lemma whose routine proof is omitted.

Lemma 2.1. *Let $\mathcal{N}$ be a reticulation-visible network on X. If $|X| = 1$, then $\mathcal{N}$ consists of the single vertex in X.*

3. The Algorithm

In this section, we present the algorithm TreeDetection. This algorithm takes as input a reticulation-visible network $\mathcal{N}$ on X and a rooted binary phylogenetic X-tree $\mathcal{T}$ and, as we establish in this section, outputs **Yes** if $\mathcal{N}$ displays $\mathcal{T}$ and **No** if $\mathcal{N}$ does not display $\mathcal{T}$. We begin with some further preliminaries.

Let $\mathcal{N}$ be a phylogenetic network on X with root ρ. Let a and b be distinct elements in X. We define the vertex v_a of $\mathcal{N}$ to be the reticulation at minimum path length from ρ such that a verifies v_a and no other element of X verifies v_a. If there is no such reticulation, we define v_a to be a. Furthermore, we define ρ_{ab} to be the vertex at maximum path length from ρ such that both a and b verify ρ_{ab}. Note that v_a and ρ_{ab} are both well defined since if $A \subseteq X$ and each element of A verifies vertices u and v of $\mathcal{N}$, then it must be that either u is an ancestor of v or v is an ancestor of u.

Briefly, TreeDetection proceeds by picking a cherry $\{a, b\}$ of the targeted rooted binary phylogenetic tree $\mathcal{T}$ and then considers how the leaves a and b are related in $\mathcal{N}$. There are various cases to consider, but in most cases we can either declare **No** directly, or we find an arc of $\mathcal{N}$ that can be deleted so that the resulting phylogenetic network displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. In the remaining cases, we can delete the leaf b from both $\mathcal{N}$ and $\mathcal{T}$ so that the resulting phylogenetic network displays the resulting rooted binary phylogenetic tree if and only if $\mathcal{N}$ displays $\mathcal{T}$. In any case, if we cannot immediately declare **No**, we reduce the size of $\mathcal{N}$ by either a vertex or an arc and, by iterating the procedure, we eventually reduce the problem to the trivial case $|X| = 1$.

The algorithm TreeDetection consists of a number of subroutines. These subroutines are partitioned into three types. The first type consists of 7 subroutines which we refer to as the *easy cases*. The other two types are referred to as *special cases*. The second type consists of 5 subroutines and the third type consists of 4 subroutines. We first give a top-level description of TreeDetection before detailing the cases and their subroutines.

Algorithm 1 TreeDetection$(\mathcal{N}, \mathcal{T})$

Input: A reticulation-visible network $\mathcal{N}$ on X and a rooted binary phylogenetic X-tree $\mathcal{T}$.
Output: **Yes** if $\mathcal{N}$ displays $\mathcal{T}$, and **No** otherwise.

if $|X| = 1$ **then**
  Return **Yes**
else
  Find a cherry $\{a, b\}$ in $\mathcal{T}$ at maximum path length from the root.
  for $i = 1$ to 7 **do**
    if Easy Case i applies **then**
      Execute subroutine Easy Case i.
      Halt.
  if the sibling of the parent of a and b in $\mathcal{T}$ is a single leaf c **then**
    for $i = 1$ to 4 **do**
      if Special Case 1.i applies **then**
        Execute subroutine Special Case 1.i.
        Halt.
    Execute subroutine Special Case 1.5.
  else the sibling of the parent of a and b in $\mathcal{T}$ is the parent of a cherry $\{c, d\}$
    for $i = 1$ to 3 **do**
      if Special Case 2.i applies **then**
        Execute subroutine Special Case 2.i.
        Halt.
    Execute subroutine Special Case 2.4.

With the top-level description of the algorithm established, we now turn to the details of the various cases. If $|X| > 1$, let $\{a, b\}$ be a cherry in $\mathcal{T}$ whose distance from its root is maximised. We then have the following 'easy' cases (each one holding only if the preceeding cases do not hold). Subroutines for each easy case as well as their justifications are given in Section 3.1. Illustrations of the last four easy cases, (EC4)–(EC7), are given in Fig. 2.

Easy Cases

(EC1) The 2-element subset $\{a, b\}$ is a cherry in $\mathcal{N}$.
(EC2) For some $i \in \{a, b\}$, we have $v_i = i$.
(EC3) For $\{i, j\} = \{a, b\}$, we have v_i is an ancestor of v_j.
(EC4) For $\{i, j\} = \{a, b\}$, there is an ancestor w of v_i that is verified by i but not by j.

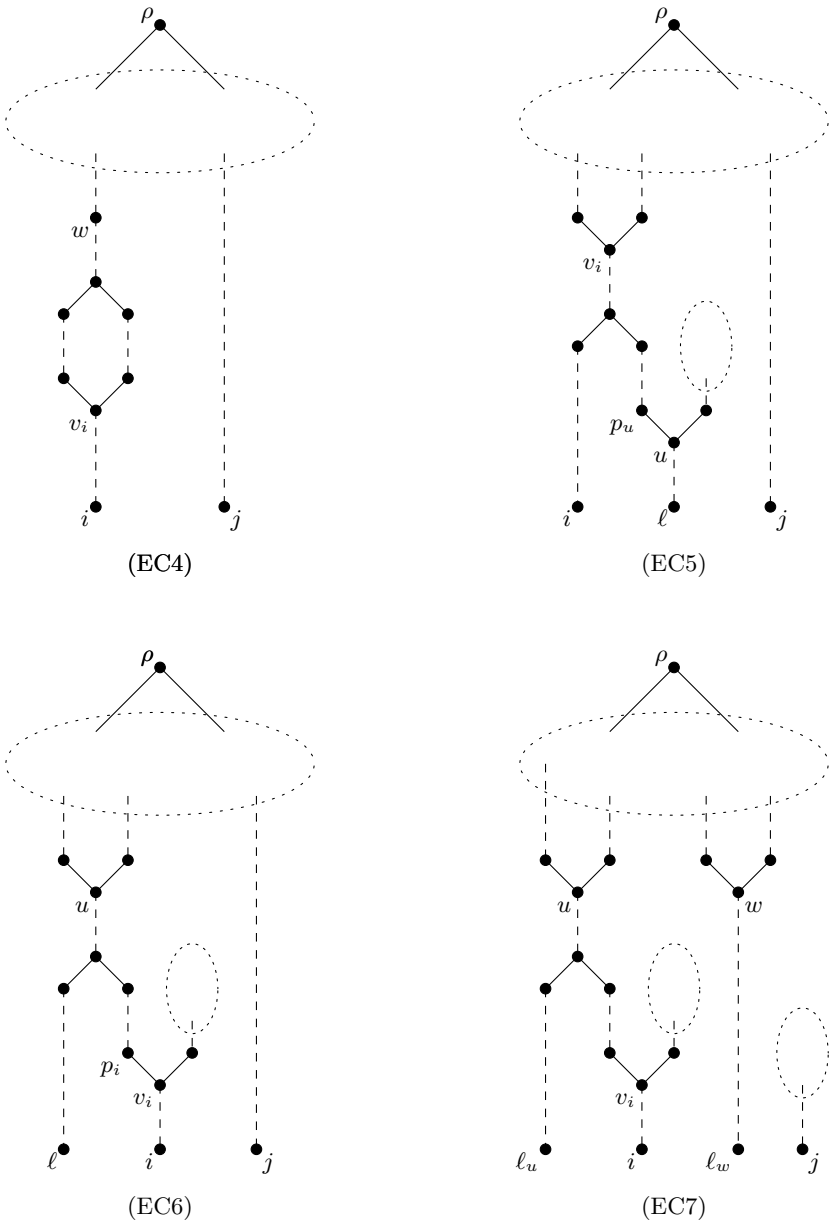
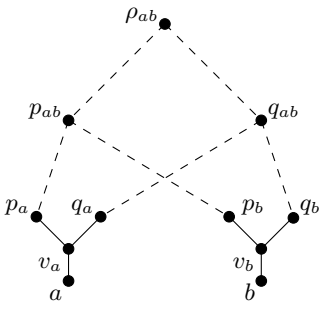
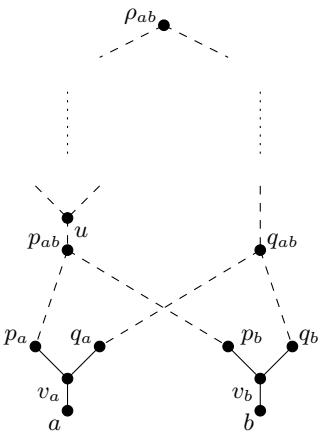


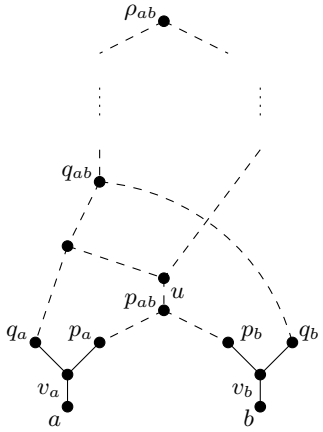
FIGURE 2. Illustrations of (EC4)–(EC7). Dashed lines between vertices correspond to (directed) paths, while solid lines correspond to edges. Only paths guaranteed by the assumptions of the particular case are shown. Note that in (EC7) leaf j is also a descendant of u and w, although exactly where the paths to j branch off the structure below u and w shown in the figure is not determined.



(i) Situation 1



(ii) Situation 2



(iii) Situation 3

FIGURE 3. Situations 1, 2, and 3.

(EC5) For some $i \in \{a, b\}$, there is a descendant u of v_i that is a reticulation not verified by i.
(EC6) For $\{i, j\} = \{a, b\}$, there is a reticulation u that is an ancestor of v_i but not of j.
(EC7) For some $i \in \{a, b\}$, there are two non-comparable reticulations u and w that are descendants of ρ_{ab} and ancestors of v_i.

We establish in Lemma 3.1 below that if we are not in one of (EC1)–(EC7), then we must be in one of the three situations illustrated in Fig. 3. Here, Figs 3(i)–(iii) is a subgraph of the phylogenetic network at the completion of (EC1)–(EC7). More precisely, each of these subgraphs consists of all (directed) paths from ρ_{ab} to either a or b. Solid lines correspond to edges.

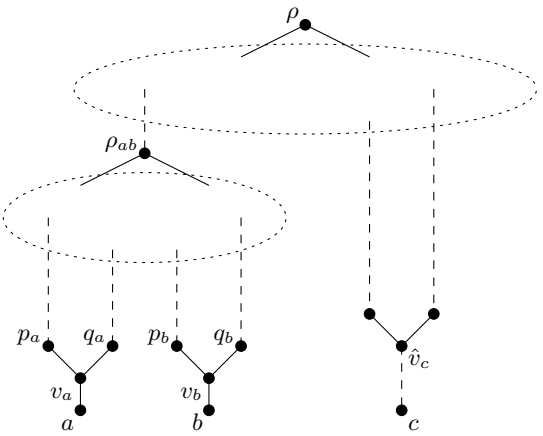
Dashed lines between named vertices correspond to (directed) paths with no reticulations unless one of the named vertices is u. Note that v_a and v_b are the parents of a and b, respectively. In Situations 2 and 3, the structure of the subgraph immediately 'below' ρ_{ab} is not completely determined. Two key vertices in the analysis to follow are those labelled p_{ab} and q_{ab} in Fig. 3. If there is a reticulation on any path between ρ_{ab} and a parent of v_a, then there is a unique maximal reticulation u by (EC7) as each pair of such reticulations are comparable. In this case we define p_a to be the parent of v_a on the path from u to v_a and q_a to be the other parent of v_a. If there is no reticulation, let the children of ρ_{ab} be p and q, and define p_a to be the parent of v_a on the path from p to v_a, q_a to be the parent of v_a on the path from q to v_a. In each case, define p_b, q_b likewise. Then p_{ab} is the ancestor of p_a and p_b at maximal distance from ρ_{ab} and q_{ab} is the ancestor of q_a and q_b at maximal distance from ρ_{ab}. (The proof of Lemma 3.1 shows that these are well defined.)

For each situation, the crucial feature we will need to consider is the structure outside of the cherry $\{a, b\}$ in $\mathcal{T}$. Since $\{a, b\}$ was chosen to be at maximum distance from the root, the sibling of the parent of a and b in $\mathcal{T}$ is either a leaf c or the parent of leaves c and d, where $\{c, d\}$ is a cherry. First, we consider the case that the sibling is a single leaf c. Let v_c be the reticulation in $\mathcal{N}$ at minimum path length from the root such that c verifies v_c and no other element of X verifies v_c. If there is no such reticulation, define v_c to be c. Let $\hat{v}_c$ be the first reticulation on a path from v_c to c (including v_c itself) that is not an ancestor of a or b. Note that $\hat{v}_c = c$ if no such vertex exists. Furthermore, c verifies $\hat{v}_c$, otherwise some other leaf would verify it, and this leaf would also verify v_c. The first of the special cases is detailed below. The corresponding subroutines and their justifications are given in Section 3.2. Illustrations of (SC1.1)–(SC1.4) are given in Figs. 4 and 5.

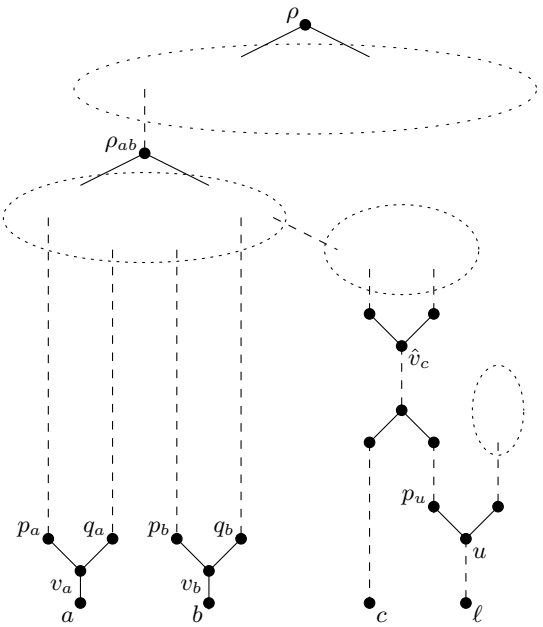
Special Case: Single Leaf Sibling

(SC1.1) The leaf c is not a descendant of ρ_{ab}.
(SC1.2) There is a descendant u of $\hat{v}_c$ that is a reticulation and not verified by c.
(SC1.3) There is an ancestor u of $\hat{v}_c$ that is a reticulation and not an ancestor of a or b.
(SC1.4) The leaf c does not verify ρ_{ab}.

The fifth subroutine deals with the case when none of (SC1.1)–(SC1.4) applies. If $\hat{v}_c = c$, denote the parent of c as p_c; otherwise, denote the parents of $\hat{v}_c$ as p_c and q_c. Let p'_c be the unique vertex at maximal distance from ρ_{ab} that is an ancestor of p_c and an ancestor of either a or b. Likewise, let q'_c be the unique vertex at maximal distance from ρ_{ab} that is an ancestor of q_c and



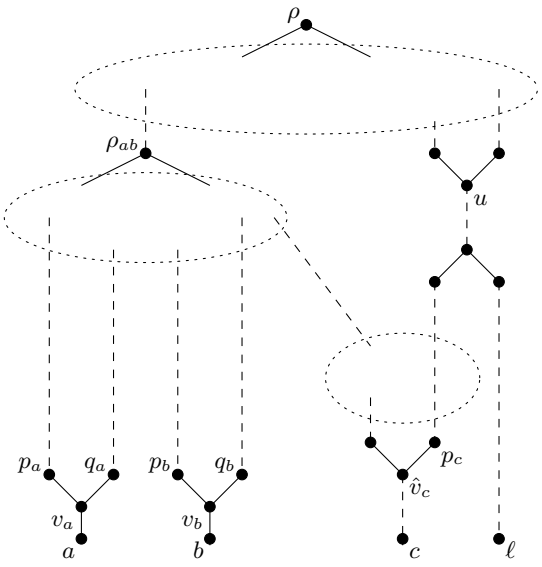
(SC1.1)



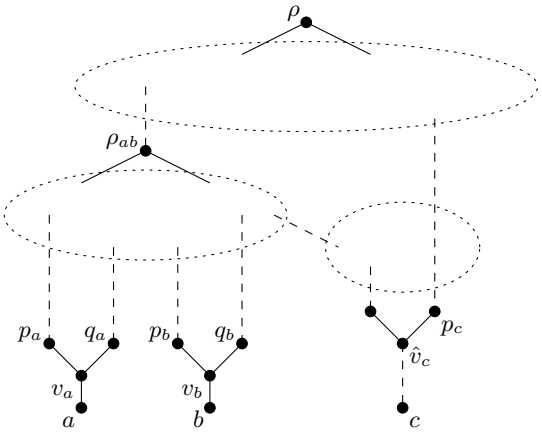
(SC1.2)

FIGURE 4. Cases (SC1.1) and (SC1.2). Dashed lines between vertices correspond to paths, while solid lines correspond to edges. Only paths guaranteed by the case are shown.

an ancestor of either a or b. For each of p'_c and q'_c either it is an ancestor of p_{ab} or an ancestor of q_{ab}, or it lies on one of the following (directed) paths: p_{ab} to p_a, call it P_1; p_{ab} to p_b, call it P_2; q_{ab} to q_a, call it P_3; and q_{ab} to q_b, call it P_4. Fig. 6 shows a possible instance of (SC1.5) when $\hat{v}_c \neq c$ and Situation 1 holds; in this case p'_c is an ancestor of p_{ab} and q'_c is on path P_4.



(SC1.3)



(SC1.4)

FIGURE 5. Cases (SC1.3) and (SC1.4). Dashed lines between vertices correspond to paths, while solid lines correspond to edges. Only paths guaranteed by the case are shown.

With the above definitions, we identify an arc to delete from $\mathcal{N}$, or a leaf to delete from both $\mathcal{T}$ and $\mathcal{N}$. The appropriate deletions are given in Tables 1 and 2 depending on whether or not $\hat{v}_c = c$, while justification for these deletions is given in Section 3.2. In the tables, we denote that p'_c (respectively, q'_c) is an ancestor of a vertex u by $> u$.

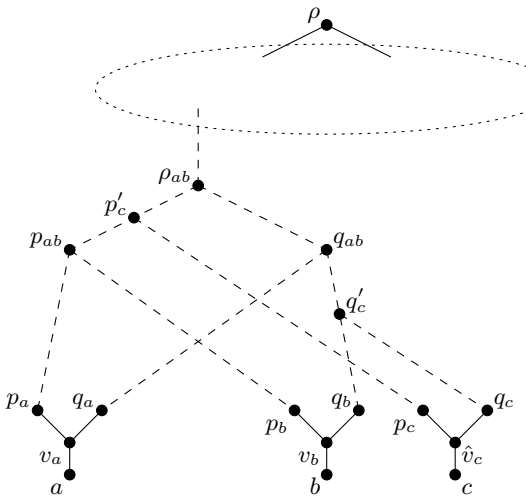


FIGURE 6. An instance of (SC1.5) when $\hat{v}_c \neq c$ and Situation 1 holds. Dashed lines correspond to paths and solids lines correspond to edges.

TABLE 1. Table showing the actions to perform on $\mathcal{N}$ to create $\mathcal{N}'$ depending on the location of p'_c in Situations 1–3 when the sibling of the parent of a and b in $\mathcal{T}$ is c and $\hat{v}_c = c$. The entry corresponding to the subcase we are in contains a single arc e, and $\mathcal{N}' = \mathcal{N} \backslash e$ and $\mathcal{T}' = \mathcal{T}$.

p'_c	Sit. 1–3
P_1 or P_2	(p_a, v_a)
P_3 or P_4	(q_a, v_a)
$> p_{ab}$	(q_a, v_a)
$> q_{ab}$	(p_a, v_a)

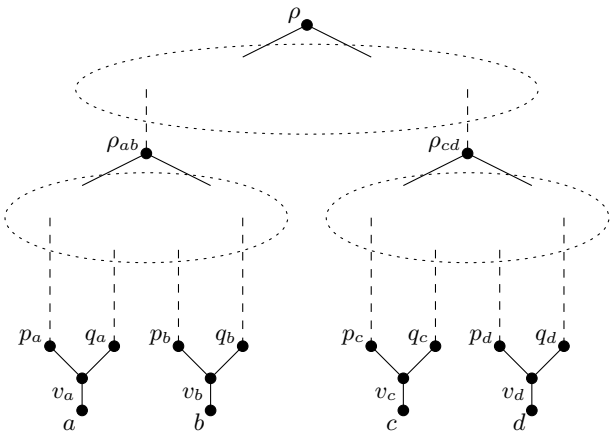
TABLE 2. Table showing the actions to perform on $\mathcal{N}$ and $\mathcal{T}$ to create $\mathcal{N}'$ and $\mathcal{T}'$ depending on the locations of p'_c and q'_c in each of Situations 1–3 when the sibling of the parent of a and b in $\mathcal{T}$ is a single leaf c and $\hat{v}_c \neq c$. If the entry corresponding to the subcase we are in contains a single arc e, then $\mathcal{N}' = \mathcal{N}\backslash e$ and $\mathcal{T}' = \mathcal{T}$. If the entry contains the leaf b, then $\mathcal{N}' = \mathcal{N}\backslash b$ and $\mathcal{T}' = \mathcal{T}\backslash b$.

p'_c	q'_c	Sit. 1	Sit. 2	Sit. 3
P_1 or P_2	P_1 or P_2	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
P_1 or P_2	P_3 or P_4	$(p_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$
P_1 or P_2	$> p_{ab}$	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$
P_1 or P_2	$> q_{ab}$	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
P_3 or P_4	P_1 or P_2	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$
P_3 or P_4	P_3 or P_4	(q_a, v_a)	(q_a, v_a)	(q_a, v_a)
P_3 or P_4	$> p_{ab}$	(q_a, v_a)	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$
P_3 or P_4	$> q_{ab}$	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$	$(p_c, \hat{v}_c)$
$> p_{ab}$	P_1 or P_2	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$
$> p_{ab}$	P_3 or P_4	(q_a, v_a)	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$
$> p_{ab}$	$> p_{ab}$	(q_a, v_a)	(q_a, v_a)	(q_a, v_a)
$> p_{ab}$	$> q_{ab}$	b	b	b
$> q_{ab}$	P_1 or P_2	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
$> q_{ab}$	P_3 or P_4	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$	$(q_c, \hat{v}_c)$
$> q_{ab}$	$> p_{ab}$	b	b	b
$> q_{ab}$	$> q_{ab}$	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)

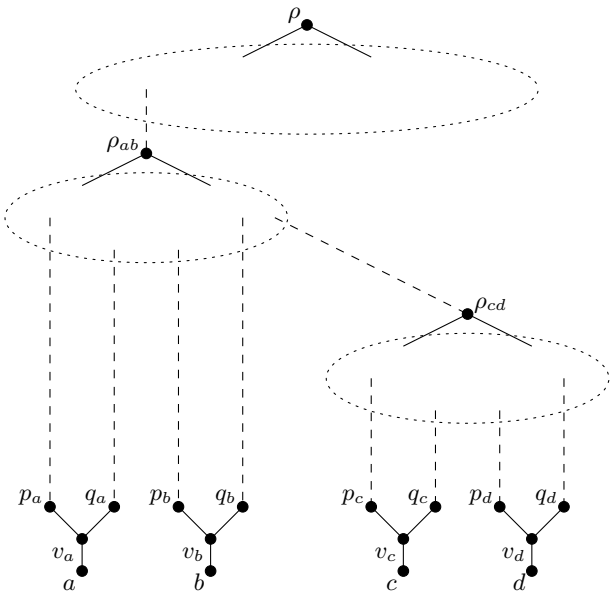
Finally, we consider the case that the sibling of the parent of a and b in $\mathcal{T}$ is the parent of leaves c and d, where $\{c, d\}$ is a cherry. Since $\{c, d\}$ is also a cherry at maximum distance from the root of $\mathcal{T}$, we first check whether any of (EC1)–(EC7) apply when considering $\{c, d\}$ instead of $\{a, b\}$. If so, we make the appropriate reduction of $\mathcal{N}$; otherwise, we may assume that the configuration of $\{c, d\}$ in $\mathcal{N}$ is analogous to one of Situations 1–3. In particular, v_c is the parent of c and v_d is the parent of d. Furthermore, without loss of generality, we may assume that ρ_{ab} is an ancestor of (or equal to) ρ_{cd} or that they are non-comparable (if this is not the case, a simple relabelling will resolve the issue). The second special case is detailed below. The corresponding subroutines and their justifications are given in Section 3.2. Cases (SC2.1) and (SC2.2) are illustrated in Fig. 7, and (SC2.3) when u is an ancestor of q_{cd} is illustrated in Fig. 8.

SPECIAL CASE: CHERRY SIBLING

(SC2.1) The vertices ρ_{cd} and ρ_{ab} are non-comparable.
(SC2.2) The vertex ρ_{cd} is not on one of the paths from ρ_{ab} to either v_a or v_b.



(SC2.1)

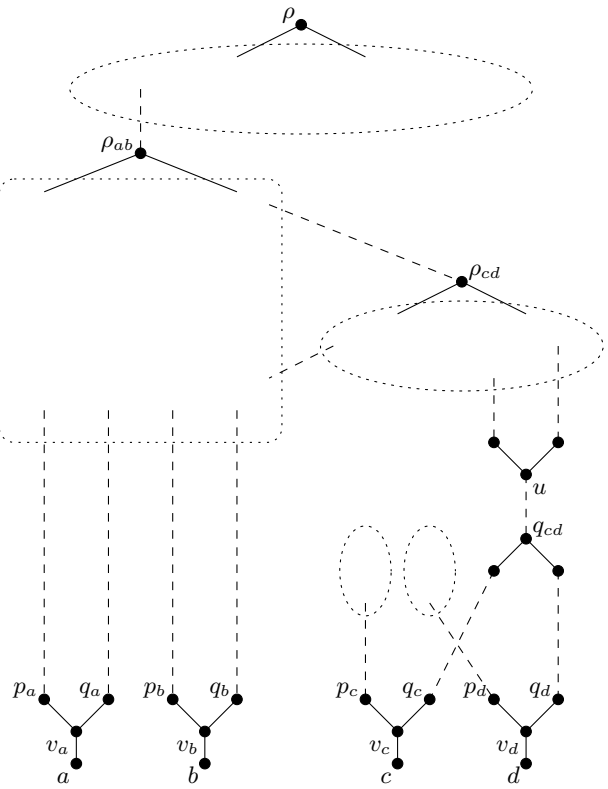


(SC2.2)

FIGURE 7. Cases (SC2.1) and (SC2.2). Dashed lines between vertices correspond to paths and solid lines correspond to edges. Only paths guaranteed by the cases are shown.

(SC2.3) There is a reticulation u that is an ancestor of p_{cd} or q_{cd} but it is not an ancestor of a or b.

Now we consider the locations of p_{cd} and q_{cd} relative to a and b. Since ρ_{cd} is a descendant of ρ_{ab} (or $\rho_{cd} = \rho_{ab}$), each of p_{cd} and q_{cd} must either lie on a path from ρ_{ab} to either v_a or v_b, or have a unique ancestor on such a path at



(SC2.3)

FIGURE 8. Case (SC2.3) when u is an ancestor of q_{cd}. Dashed lines between vertices correspond to paths and solid lines correspond to edges. Only paths guaranteed by the case are shown.

maximal distance from ρ_{ab}. Let p'_{cd} be this unique ancestor of p_{cd}, or equal to p_{cd} if it lies on such a path itself, and let q'_{cd} be the analogous vertex for q_{cd}. We have a similar table to that of the previous special case, but this time we must distinguish between subcases where p'_{cd} is or is not equal to p_{cd}, and q'_{cd} is or is not equal to q_{cd}. We use the following definitions:

(i) Let p'_{cd} (respectively, q'_{cd}) be of *Type 1* if it is a descendant of p_{ab} or an ancestor of p_{ab} and $p'_{cd} = p_{cd}$ (respectively, $q'_{cd} = q_{cd}$).
(ii) Let p'_{cd} (respectively, q'_{cd}) be of *Type 2* if it is a descendant of q_{ab} or an ancestor of q_{ab} and $p'_{cd} = p_{cd}$ (respectively, $q'_{cd} = q_{cd}$).
(iii) Let p'_{cd} (respectively, q'_{cd}) be of *Type 3* if it is an ancestor of p_{ab} and $p'_{cd} \neq p_{cd}$ (respectively, $q'_{cd} \neq q_{cd}$).
(iv) Let p'_{cd} (respectively, q'_{cd}) be of *Type 4* if it is an ancestor of q_{ab} and $p'_{cd} \neq p_{cd}$ (respectively, $q'_{cd} \neq q_{cd}$).

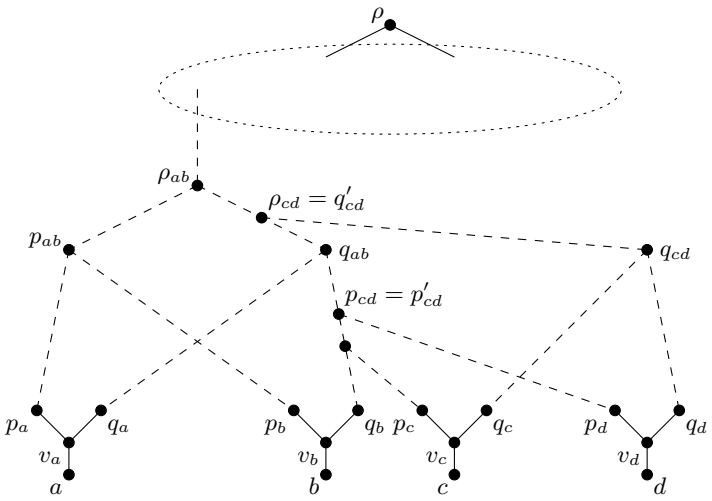


FIGURE 9. An instance of (SC2.4). Here p'_{cd} is of Type 2, q'_{cd} is of Type 4, and Situation 1 holds for both cherries $\{a, b\}$ and $\{c, d\}$. Dashed lines between vertices correspond to paths, and solid lines correspond to edges.

As an example, a particular instance of (SC2.4) is given in Fig. 9. Here, p'_{cd} is of Type 2, q'_{cd} is of Type 4, and Situation 1 holds for both cherries.

TABLE 3. Table showing the actions to perform on $\mathcal{N}$ and $\mathcal{T}$ to form $\mathcal{N}'$ and $\mathcal{T}'$, respectively, depending on the locations of p_{cd} and q_{cd} in each of Situations 1–3 in the case that the sibling of the parent of a and b realises a cherry $\{c, d\}$. If the entry corresponding to the subcase we are in contains a single arc e, then $\mathcal{N}' = \mathcal{N}\backslash e$ and $\mathcal{T}' = \mathcal{T}$. If the entry contains the leaf b, then $\mathcal{N}' = \mathcal{N}\backslash b$ and $\mathcal{T}' = \mathcal{T}\backslash b$.

p'_{cd}	q'_{cd}	Sit. 1	Sit. 2	Sit. 3
Type 1	Type 1	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
Type 1	Type 2	(p_c, v_c)	(q_c, v_c)	(q_c, v_c)
Type 1	Type 3	(p_c, v_c)	(p_c, v_c)	(p_c, v_c)
Type 1	Type 4	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
Type 2	Type 1	(p_c, v_c)	(p_c, v_c)	(p_c, v_c)
Type 2	Type 2	(q_a, v_a)	(q_a, v_a)	(q_a, v_a)
Type 2	Type 3	(q_a, v_a)	(p_c, v_c)	(p_c, v_c)
Type 2	Type 4	(p_c, v_c)	(p_c, v_c)	(p_c, v_c)
Type 3	Type 1	(q_c, v_c)	(q_c, v_c)	(q_c, v_c)
Type 3	Type 2	(q_a, v_a)	(q_c, v_c)	(q_c, v_c)
Type 3	Type 3	(q_a, v_a)	(q_a, v_a)	(q_a, v_a)
Type 3	Type 4	b	b	b
Type 4	Type 1	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)
Type 4	Type 2	(q_c, v_c)	(q_c, v_c)	(q_c, v_c)
Type 4	Type 3	b	b	b
Type 4	Type 4	(p_a, v_a)	(p_a, v_a)	(p_a, v_a)

3.1. **Easy Cases.** Here we present the subroutines for each of (EC1)–(EC7) with justification following each subroutine. Throughout the presentation of these subroutines, as well as those presented for the two special cases, whenever we delete an arc or a leaf from $\mathcal{N}$ or $\mathcal{T}$, we additionally contract any resulting degree-two vertex.

Subroutine 1 EASY CASE 1

1: Delete b in $\mathcal{N}$ and $\mathcal{T}$ to obtain $\mathcal{N}'$ and $\mathcal{T}'$, respectively
2: Return TREEDETECTION$(\mathcal{N}', \mathcal{T}')$.

Suppose $\{a, b\}$ is a cherry in $\mathcal{N}$. Then any embedding of $\mathcal{T}'$ in $\mathcal{N}'$ can be extended to an embedding of $\mathcal{T}$ in $\mathcal{N}$ by appending b to the arc incident with a in both $\mathcal{N}'$ and $\mathcal{T}'$, and including the new edge in the embedding. Conversely, any embedding of $\mathcal{T}$ in $\mathcal{N}$ gives rise to an embedding of $\mathcal{T}'$ in $\mathcal{N}'$ by deleting b, its incident arc, and contracting the resulting degree-two vertex.

Subroutine 2 EASY CASE 2

Requirement: Easy Cases 1 does not apply, and $v_i = i$ for some $i \in \{a, b\}$.

1: Let p_i denote the parent of i.
2: **if** there is a tree path from p_i to a leaf $\ell \notin \{a, b\}$, **then**
3: Return **No**.
4: **else** there is a tree path from p_i to a parent p_u of a reticulation u.
5: **if** $u \neq v_j$, **then**
6: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_u, u).
7: **else** $u = v_j$
8: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to u that is not (p_u, u).
9: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

If there is a tree path from p_i to a leaf $\ell \notin \{a, b\}$, then every rooted binary phylogenetic X-tree embedded in $\mathcal{N}$ must display the rooted triple $i\ell|j$. Since $\{i, j\} = \{a, b\}$ is a cherry in $\mathcal{T}$, no embedding of $\mathcal{T}$ in $\mathcal{N}$ can exist. Thus returning **No** in Line 3 is correct.

So assume that there is no such tree path from p_i. Then there is a tree path from p_i to a parent p_u of a reticulation u. Assume $u \neq v_j$. Observe that u is not a descendant of v_j, otherwise i would also verify v_j contradicting the definition of v_j. Thus u is verified by some leaf $\ell \notin \{a, b\}$. Note that we can choose $\ell \neq j$ since $u \neq v_j$ and u is not a descendant of v_j. Now an embedding of a rooted binary phylogenetic X-tree in $\mathcal{N}$ that uses arc (p_u, u) would display the rooted triple $i\ell|j$ if v_j is not a descendant of u, and $j\ell|i$ if v_j is a descendant of u.

On the other hand, if $u = v_j$, then it is easily seen that if there is an embedding of $\mathcal{T}$ that does not use arc (p_u, u), then there is also an embedding of $\mathcal{T}$ that uses arc (p_u, u). Thus in both cases there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

The justifications for the next three subroutines are similar to that for Subroutine 2 and are omitted.

Subroutine 3 EASY CASE 3

Requirement: Easy Cases 1–2 do not apply, and v_i is an ancestor of v_j for $\{i, j\} = \{a, b\}$.

Let p_j be a parent of v_j that is a descendant of v_i, and let u be the first reticulation reached on a backward path from p_j to v_i.
if u is not an ancestor of i, **then**
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_j, v_j).
else u is an ancestor of i
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to v_j that is not (p_j, v_j).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Subroutine 4 EASY CASE 4

Requirement: Easy Cases 1–3 do not apply and, for some $i \in \{a, b\}$, there is an ancestor w of v_i which is verified by i but not by j.

if w is not an ancestor of j, **then**
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting either incoming arc to v_i.
else if there is a reticulation on a path P from w to v_j not equal to v_j, that is not an ancestor of i, **then**
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to v_j that is on P.
else
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to v_j that is not on a backward path to w from v_j.
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Subroutine 5 EASY CASE 5

Requirement: Easy Cases 1–4 do not apply and, for some $i \in \{a, b\}$, there is a path P from v_i to a reticulation not verified by i.

Let u be the first reticulation on P not verified by i and let p_u be the parent of u on P.
Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_u, u).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Subroutine 6 EASY CASE 6

Requirement: Easy Cases 1–5 do not apply and, for $\{i, j\} = \{a, b\}$, there is a reticulation u that is an ancestor of v_i but not an ancestor of j.

Let p_i be a parent of v_i that is a descendant of u.
Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_i, v_i).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Since u is an ancestor of v_i it must be verified by some leaf $\ell \neq i$. But u is not an ancestor of j, so $\ell \neq j$. If an embedding of $\mathcal{T}$ uses the arc (p_i, v_i), then it must display the rooted triple $i\ell|j$; a contradiction. Thus there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

Subroutine 7 EASY CASE 7

Requirement: Easy Cases 1–6 do not apply, and there are two non-comparable reticulations u and w that are descendants of ρ_{ab} and, for some $i \in \{a, b\}$, ancestors of v_i.

Let p_i be a parent of v_i that is a descendant of u, and let q_i be the other parent of v_i.
Let ℓ_u be a leaf verifying u, and let ℓ_w be a leaf verifying w.
if there is a reticulation u' that is on a path from u to v_i at minimal distance from u **then**
    Let $\ell_{u'} \notin \{\ell_u, \ell_w, a, b\}$ be a leaf verifying u'.
    if $\mathcal{T}$ displays the rooted triple $\ell'_u\ell_u|\ell_w$ **then**
        Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to u' on the path from w.
    else
        Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the incoming arc to u' on the path from u.
else
    if $\mathcal{T}$ displays the rooted triple $i\ell_u|\ell_w$ **then**
        Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (q_i, v_i).
    else
        Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_i, v_i).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Without loss of generality, we may assume u and w are at maximal distance from ρ_{ab}, that is, there is no reticulation u' that is a descendant of u such that u' and w satisfy the conditions of the requirement and, likewise, no reticulation w' that is a descendant of w such that u and w' satisfy the conditions of the requirement. Since u and w are descendants of ρ_{ab}, neither is verified by both a and b and so, by (EC4), neither is verified by a or by b. Thus there are leaves $\ell_u, \ell_w \notin \{a, b\}$ such that ℓ_u verifies u and ℓ_w verifies w. Since u and w are non-comparable, $\ell_u \neq \ell_w$ as not all paths to a single leaf can go through two non-comparable vertices. By (EC6), each of u and w is an ancestor of both a and b.

Suppose first that there is a reticulation u' that is on a path from u to a parent of v_i. Take u' to be such a reticulation at shortest distance to u. By our assumption, u' cannot be non-comparable with w, so it must be a descendant of both w and u. Furthermore, u' must be verified by some

$\ell_{u'} \notin \{\ell_u, \ell_w, a, b\}$. Since $\mathcal{T}$ must display at most one of $\ell'_u\ell_u|\ell_w$ and $\ell'_u\ell_w|\ell_u$; we delete the appropriate reticulation arc incident with u' so that there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

By symmetry, we may now assume that there is no reticulation that is on a path from u to v_i or from w to v_i. Let p_i be the parent of v_i that is a descendant of u, and let q_i be the parent of v_i that is a descendant of w. Note that these are well defined by our latest assumption and, also, $p_i \neq q_i$ since u and w are non-comparable. If an embedding of $\mathcal{T}$ uses arc (p_i, v_i), it must display the rooted triple $i\ell_u|\ell_w$, and if it uses arc (q_i, v_i), it must display the rooted triple $i\ell_w|\ell_u$. At most one of these rooted triples is displayed by $\mathcal{T}$. Thus, with $\mathcal{N}'$ as constructed, there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

3.2. **Special Cases.** We first show that if none of (EC1)–(EC7) hold, then we are indeed in one of Situations 1, 2, and 3. Thereafter, we present the subroutines and their justifications for the special cases.

Lemma 3.1. *Let $|X| \geq 2$, and let $\mathcal{N}$ be a phylogenetic network on X and let $\mathcal{T}$ be a rooted binary phylogenetic X-tree. Let $\{a, b\}$ be a cherry in $\mathcal{T}$ whose distance from its root is maximised. If none of* (EC1)–(EC7) *applies, then one of Situations* 1, 2, *and* 3 *applies.*

Proof. Let $i \in \{a, b\}$. Suppose first that the parent of i is a tree vertex v. Then there is a tree-path from v that reaches either a parent of a reticulation u not verified by i, or a leaf $\ell \neq i$. By (EC2), $v_i \neq i$, so (EC5) excludes the tree-path reaching a parent of a reticulation u. But then ℓ verifies every vertex that i verifies (except i itself), contradicting the definition of v_i.

Now suppose that the parent of i is a reticulation $w \neq v_i$. Let the parents of w be p_w and q_w, and note that p_w and q_w are both tree vertices since parents of reticulations are not visible. The vertex v_i must be an ancestor of both p_w and q_w. By (EC5), p_w cannot have a path to a reticulation other than w. But then p_w must have a tree path to a leaf ℓ that verifies v_i; a contradiction. It now follows that v_i is the parent of i for all $i \in \{a, b\}$.

The vertex ρ_{ab} cannot be a reticulation, as its child would also be verified by a and b. Thus ρ_{ab} has two children p and q, neither of which is verified by both a and b. Furthermore, by (EC4), neither p nor q is verified by exactly one of a and b. Hence there is a (directed) path from p to each of a and b, and also from q to each of a and b.

Except for v_a and v_b, suppose that there are no reticulations on any path between ρ_{ab} and v_a and between ρ_{ab} and v_b. Then the paths from p to each of a and b are unique, and the paths from q to each of a and b are unique.

Denote the parent of v_a on the path from p to a by p_a, and the parent of v_a on the path from q to a by q_a. Similarly, denote by p_b and q_b the parents of v_b. Let p_{ab} be the last vertex on the path from p to p_a that is an ancestor of b, and let q_{ab} be the last vertex on the path from q to q_a that is an ancestor of b. (Possibly $p_{ab} = p$ or $q_{ab} = q$.) Thus, if, apart from v_a and v_b, there are no reticulations on any path between ρ_{ab} and a, and between ρ_{ab} and b, then we are in Situation 1.

Now suppose that, in addition to v_a and v_b, there is a reticulation on a path between ρ_{ab} and v_a or between ρ_{ab} and v_b. Let u be such a reticulation at maximal distance from ρ_{ab}. By (EC6), u is an ancestor of both a and b. By (EC7), every other reticulation on a path from ρ_{ab} to v_a or from ρ_{ab} to v_b that is not v_a or v_b is an ancestor of u. By maximality, the paths from u to a and from u to b are unique. Let the parent of v_a on the path from u to a be p_a and let the parent of v_b on the path from u to b be p_b. Let p_{ab} be the last vertex on the path from u to p_a that is an ancestor of b. Let q_a and q_b be the other parents of v_a and v_b, respectively. Let q_{ab} be the ancestor of q_a and q_b at maximal distance from ρ_{ab}. Note that q_{ab} is well defined. To see this, assume that there were two possibilities q_{ab} and q'_{ab}, both ancestors of q_a and q_b, and at maximal distance from ρ_{ab}. By maximality, q_{ab} and q'_{ab} are non-comparable, so there would have to be a reticulation where the paths from q_{ab} and q'_{ab} to q_a merge, and where the paths from q_{ab} and q'_{ab} to q_b merge. These two reticulations would also have to be non-comparable, contradicting the exclusion of (EC7).

If q_{ab} is not an ancestor of u, then we are in Situation 2. If q_{ab} is an ancestor of u, then we are in Situation 3. □

We now present the subroutines for the Special Case when the sibling of the parent of a and b in $\mathcal{T}$ is a single leaf c.

Subroutine 8 SPECIAL CASE 1.1

Requirement: The leaf c is not a descendant of ρ_{ab}.
1: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_a, v_a).
2: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

This is valid since any embedding of $\mathcal{T}$ in $\mathcal{N}$ must have the last common ancestor of a and c as an ancestor of ρ_{ab}. Thus there are no paths (in the embedding) from ρ_{ab} to any leaves other than a and b, so we can resolve v_a either way without changing the result of the embedding. In particular, $\mathcal{N}$ displays $\mathcal{T}$ if and only if $\mathcal{N}'$ displays $\mathcal{T}$.

The justification of the next subroutine is similar to that given for EASY CASE 2 and is omitted.

Subroutine 9 SPECIAL CASE 1.2

Requirement: There is a path P from $\hat{v}_c$ to a reticulation not verified by c.

1: Let u be the first reticulation on P not verified by c and let p_u be the parent of u on P.
2: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_u, u).
3: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Subroutine 10 SPECIAL CASE 1.3

Requirement: There is an ancestor u of $\hat{v}_c$ that is a reticulation and not an ancestor of a or b.

1: Let p_c be a parent of $\hat{v}_c$ that is a descendant of u
2: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc $(p_c, \hat{v}_c)$.
3: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Since u is an ancestor of $\hat{v}_c$, but not an ancestor of a or b, we must have $\hat{v}_c = v_c$, and so u must be verified by some leaf $\ell \neq c$. Since u is not an ancestor of a or b, it follows that $\ell \notin \{a, b\}$. If an embedding of $\mathcal{T}$ uses the arc $(p_c, \hat{v}_c)$, then it must display the rooted triple $c\ell|a$; a contradiction. Thus there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

Subroutine 11 SPECIAL CASE 1.4

Requirement: The leaf c does not verify ρ_{ab}.

1: Let p_c be a parent of $\hat{v}_c$ that is not a descendant of ρ_{ab}.
2: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc $(p_c, \hat{v}_c)$.
3: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

If c does not verify ρ_{ab}, then there is a (directed) path P from an ancestor of ρ_{ab} to $\hat{v}_c$ not via ρ_{ab}, and p_c is on this path. If we are in Situation 2 or Situation 3, then there is no embedding of $\mathcal{T}$ in $\mathcal{N}$ that uses arc $(p_c, \hat{v}_c)$. Otherwise, the embedding of $\mathcal{T}$ displays the rooted triple $a\ell|c$, where ℓ is a leaf verifying u; a contradiction. If we are in Situation 1 and there is an embedding of $\mathcal{T}$ that uses arc $(p_c, \hat{v}_c)$, then in the embedding the last common ancestor of a and c is an ancestor of ρ_{ab}, so (as above) we could adjust the embedding by resolving v_a, v_b either way and it would still be valid. By (SC1.3), there is an unique path Q from ρ_{ab} to $\hat{v}_c$. Now Q cannot use both p_{ab} and q_{ab}, so we can resolve v_a and v_b towards p_a and p_b, or towards q_a and q_b, and resolve $\hat{v}_c$ towards its parent on Q, and still have a valid embedding, but one not using arc $(p_c, \hat{v}_c)$. Thus there is an embedding of $\mathcal{T}$ in $\mathcal{N}$ if and only if there is an embedding of $\mathcal{T}$ in $\mathcal{N}'$.

To complete the analysis of the first special case, we present that last of its subroutines and justify the actions in Tables 1 and 2.

Subroutine 12 SPECIAL CASE 1.5

Requirement: Easy Cases 1–7 and Special Cases 1.1–1.4 do not apply. The sibling of the parent of a and b in $\mathcal{T}$ is a single leaf c.

1: Depending on whether $\hat{v}_c = c$, determine which of Tables 1 and 2 to use and which entry applies.
2: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc listed and set $\mathcal{T}' = \mathcal{T}$, or construct $\mathcal{N}' = \mathcal{N} \backslash b$ and $\mathcal{T}' = \mathcal{T} \backslash b$ if b is listed instead of an arc.
3: Return TREEDETECTION$(\mathcal{N}', \mathcal{T}')$.

We justify the subroutine for when $\hat{v}_c \neq c$. If $\hat{v}_c = c$, the justification is simpler and omitted. If an embedding of $\mathcal{T}$ in $\mathcal{N}$ uses the arc $(p_c, \hat{v}_c)$ and p'_c is on the path P_1 or P_2, then v_a and v_b will have to resolve towards q_a and q_b. Otherwise, the embedding does not have a vertex whose leaf descendants are just a and b. Hence, if each of p'_c and q'_c lie on P_1 or P_2, then we can delete (p_a, v_a). Similarly, if each of p'_c and q'_c lie on P_3 or P_4, then we can delete (q_a, v_a).

If we are in Situation 1, and p'_c lies on P_1 or P_2 and q'_c lies on P_3 or P_4, then, in an embedding of $\mathcal{T}$ in $\mathcal{N}$, the last common ancestor of a and c would be ρ_{ab} whichever way we resolved $\hat{v}_c$. Thus, in any of these cases, there is no path to a leaf other than a, b, and c from ρ_{ab} and so, if an embedding exists, one exists with $\hat{v}_c$ resolved each way (and v_a and v_b resolved accordingly). Thus, we can delete $(p_c, \hat{v}_c)$. Similar argument holds if we are in Situation 1, and p'_c lies on P_3 or P_4 and q'_c lies on P_1 or P_2.

If we are in Situations 2 or 3, and p'_c lies on P_3 or P_4, no embedding of $\mathcal{T}$ can resolve $\hat{v}_c$ towards p_c. This follows as then v_a and v_b would have to resolve towards p_a and p_b and, if ℓ_u is a leaf that verifies u (note $\ell_u \neq c$), then the embedding of $\mathcal{T}$ in $\mathcal{N}$ would display the rooted triple $a\ell_u|c$; a contradiction. So we can delete arc $(p_c, \hat{v}_c)$. Similarly, if we are in Situations 2 or 3, and q'_c lies on P_3 or P_4, then we can delete arc $(q_c, \hat{v}_c)$.

Now suppose p'_c is an ancestor of p_{ab} (denoted $> p_{ab}$ in the table). Then, if an embedding of $\mathcal{T}$ in $\mathcal{N}$ uses the arc $(p_c, \hat{v}_c)$ and arc (q_a, v_a), the last common ancestor of a and c in the embedding would be an ancestor of p'_c. Thus there is no path to a leaf other than a, b, and c from p'_c and so an embedding exists with v_a and v_b resolved towards p_a and p_b, respectively. Similarly, if p'_c is an ancestor of q_{ab} and an embedding of $\mathcal{T}$ in $\mathcal{N}$ exists using arcs $(p_c, \hat{v}_c)$ and (p_a, v_a), then one exists with v_a and v_b resolved towards q_a and q_b, respectively. A symmetric argument holds with q'_c instead of p'_c.

Thus if p'_c is an ancestor of p_{ab} and q'_c lies on P_3 or P_4 (or vice versa), then we can delete arc (q_a, v_a) and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. Likewise, if p'_c is an ancestor of q_{ab} and q'_c lies on P_1 or P_2, then we can delete arc (p_a, v_a). Also, if both p'_c and q'_c are ancestors of p_{ab}, we can delete arc (q_a, v_a), and if both are ancestors of q_{ab}, we can delete arc (p_a, v_a). In each of the last three subcases, the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

If p'_c is an ancestor of p_{ab} and q'_c lies on P_1 or P_2, and an embedding of $\mathcal{T}$ in $\mathcal{N}$ exists that resolves $\hat{v}_c$ towards q_c, then in this embedding v_a and v_b must resolve towards q_a and q_b, respectively, in which case, in the embedding, the last common ancestor of a and c is ρ_{ab} in Situation 1, and an ancestor of u in Situations 2 and 3. In Situation 1, we could therefore resolve $\hat{v}_c$ towards p_c without changing the topology of the embedded tree. In Situations 2 and 3, if $\ell_u \neq c$, where ℓ_u is a leaf that verifies u, then the embedding displays the rooted triple $c\ell_u|a$; a contradiction. Thus $\ell_u = c$, and therefore p'_c lies on the path from u to p_{ab}. We can therefore resolve $\hat{v}_c$ towards p_c without changing the topology of the embedded tree. It follows that in either case, we can delete arc $(q_c, \hat{v}_c)$ and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. A similar argument holds if we interchange the roles of p'_c and q'_c, in which case, we delete the arc $(p_c, \hat{v}_c)$. A symmetric argument holds in Situation 1 if p'_c is an ancestor of q_{ab} and q'_c lies on P_3 or P_4 (or vice versa).

The remaining subcase is that p'_c is an ancestor of p_{ab} and q'_c is an ancestor of q_{ab}, or vice versa. We now show that $\mathcal{N}\backslash b$ displays $\mathcal{T}\backslash b$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. In any embedding of $\mathcal{T}\backslash b$ in $\mathcal{N}\backslash b$, there is a vertex whose leaf descendants are precisely a and c. Also, in the embedding, the peak of the embedded path from a to c must be an ancestor of either p_{ab} or q_{ab}, since there is no (directed) path from either to c. If this peak is an ancestor of p_{ab}, then we can extend this embedding to an embedding of $\mathcal{T}$ in $\mathcal{N}$ by adding arcs (p_b, v_b) and (v_b, b), while if it is an ancestor of q_{ab}, then we extend the embedding to such an embedding by adding arcs (q_b, v_b) and (v_b, b). These must still be valid embeddings since there is a tree path from p_{ab} to p_b, and from q_{ab} to q_b, so in the embedding b is in the correct location. Conversely, any embedding of $\mathcal{T}$ in $\mathcal{N}$ gives rise to an embedding of $\mathcal{T}\backslash b$ in $\mathcal{N}\backslash b$ by deleting b and its incoming arc. This completes the justification of Table 2.

Finally, we describe the subroutines for the special case when the sibling of the parent of a and b in $\mathcal{T}$ is the parent of a second cherry $\{c, d\}$.

Subroutine 13 SPECIAL CASE 2.1

Requirement: The vertices ρ_{cd} and ρ_{ab} are non-comparable.

1: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_a, v_a).
2: Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

If ρ_{cd} and ρ_{ab} are non-comparable, then, in any embedding of $\mathcal{T}$ in $\mathcal{N}$, the last common ancestor of a and c is an ancestor of ρ_{ab}, and so there are no paths in the embedding from ρ_{ab} to any leaf ℓ other than a and b; otherwise, the embedding displays the rooted triple $a\ell|c$. Thus, we can resolve v_a either way, and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

Subroutine 14 SPECIAL CASE 2.2

Requirement: The vertex ρ_{cd} is not on one of the paths from ρ_{ab} to either v_a or v_b.

Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_c, v_c).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

If ρ_{cd} is not on one of the paths from ρ_{ab} to either v_a or v_b, then, in any embedding of $\mathcal{T}$ in $\mathcal{N}$, the last common ancestor of a and c is an ancestor of ρ_{cd}, and so there are no paths in the embedding from ρ_{cd} to a leaf ℓ other than c and d; otherwise the embedding displays the rooted triple $c\ell|a$. Therefore we can resolve v_c either way, and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

Subroutine 15 SPECIAL CASE 2.3

Requirement: There is a reticulation u that is an ancestor of p_{cd} or q_{cd} but it is not an ancestor of a or b.

if u is an ancestor of p_{cd} **then**
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (p_c, v_c).
else
    Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc (q_c, v_c).
Return TREEDETECTION$(\mathcal{N}', \mathcal{T})$.

Reticulation u must be verified by some leaf $\ell \notin \{a, b, c, d\}$ since u is a descendant of ρ_{cd} and an ancestor of v_c and v_d, but not an ancestor of either a or b. Thus any embedding that resolves v_c towards u would display the rooted triple $c\ell|a$; a contradiction.

We now justify the final actions in Table 3.

Subroutine 16 SPECIAL CASE 2.4

Requirement: Easy Cases 1–7 and Special Cases 2.1–2.3 do not apply. The sibling of the parent of a and b in $\mathcal{T}$ is the parent of a cherry $\{c, d\}$.

1: Determine which of the entries in Table 3 applies.
2: Construct $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arc listed and set $\mathcal{T}' = \mathcal{T}$, or construct $\mathcal{N}' = \mathcal{N} \backslash b$ and $\mathcal{T}' = \mathcal{T} \backslash b$ if b is listed instead of an arc.
3: Return TREEDETECTION$(\mathcal{N}', \mathcal{T}')$.

If an embedding of $\mathcal{T}$ in $\mathcal{N}$ uses the arc (p_c, v_c) and p'_{cd} is on the path from p_{ab} to p_a or p_{ab} to p_b, then v_a and v_b will have to resolve towards q_a and q_b, respectively, so that the embedding contains a vertex whose leaf descendants are a and b. Likewise, if p'_{cd} is an ancestor of p_{ab} and $p'_{cd} = p_{cd}$, then an embedding of $\mathcal{T}$ in $\mathcal{N}$ that uses arc (p_c, v_c) implies v_a and v_b will have to resolve towards q_a and q_b, respectively; otherwise, one of the rooted triples $ac|d$ and $ad|c$ would be displayed by the embedding. The same outcomes hold if we replace (p_c, v_c) and p'_{cd} with (q_c, v_c) and q'_{cd}, respectively, in the hypotheses. Thus if both p'_{cd} and q'_{cd} are of Type 1, we can delete (p_a, v_a) since, either way v_c resolves, the embedding cannot use that arc, and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. Similarly, if both p'_{cd} and q'_{cd} are of Type 2, we can delete (q_a, v_a).

If we are in Situation 1 and, without loss of generality, p'_{cd} is Type 1 and q'_{cd} is Type 2, then, in an embedding of $\mathcal{T}$ in $\mathcal{N}$, the last common ancestor of a and c would be ρ_{ab} whichever way we resolved v_c. Thus, in any such embedding, there is no path to a leaf other than a, b, c, and d from ρ_{ab} and so, if an embedding exists, one exists with v_c resolved each way (and v_a and v_b resolved accordingly). Thus we can delete (p_c, v_c), and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

If we are in Situations 2 or 3, and p'_{cd} is of Type 2, then no embedding of $\mathcal{T}$ in $\mathcal{N}$ can resolve v_c towards p_c. This follows as then v_a and v_b would have to resolve towards p_a and p_b, respectively. But then if ℓ_u is a leaf that verifies u, and noting that $\ell_u \notin \{c, d\}$, the embedding would display the rooted triple $a\ell_u|c$; a contradiction. So we can delete arc (p_c, v_c), and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. Similarly, if q'_{cd} is of Type 2, we can delete arc (q_c, v_c).

Now suppose p'_{cd} is of Type 3. Then if an embedding of $\mathcal{T}$ in $\mathcal{N}$ uses the arc (p_c, v_c) and arc (q_a, v_a), it follows that the last common ancestor of a and c in this embedding would be an ancestor of p'_{cd}. Thus in such an embedding there is no path to a leaf other than a, b, c, and d from p'_{cd}, and so an embedding also exists with v_a and v_b resolved towards p_a and p_b. Similarly, if p'_{cd} is of Type 4 and an embedding of $\mathcal{T}$ in $\mathcal{N}$ exists using arcs

(p_c, v_c) and (p_a, v_a), then one exists with v_a and v_b resolved towards q_a and q_b. A symmetric argument holds with q'_{cd} instead of p'_{cd}.

Thus if p'_{cd} is of Type 3 and q'_{cd} is of Type 2 (or vice versa), we can delete arc (q_a, v_a), and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. Likewise, if p'_{cd} is of Type 4 and q'_{cd} is of Type 1 (or vice versa), we can delete arc (p_a, v_a). Also, if both p'_{cd} and q'_{cd} are of Type 3, we can delete arc (q_a, v_a), and if both are of Type 4, we can delete arc (p_a, v_a). In each of the last three subcases, the resulting phylogenetic network displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

Suppose that p'_{cd} is of Type 3 and q'_{cd} is of Type 1, and that an embedding of $\mathcal{T}$ in $\mathcal{N}$ exists that resolves v_c and v_d towards q_c and q_d, respectively. Then, in this embedding, v_a and v_b must resolve towards q_a and q_b, respectively, and the last common ancestor of a and c is ρ_{ab} in Situation 1, and an ancestor of u in Situation 2 and 3. In Situation 1, we could therefore resolve v_c and v_d towards p_c and p_d without changing the topology of the embedded tree. In Situations 2 and 3, if $l_u \notin \{c, d\}$, where ℓ_u is a leaf that verifies u, then the embedding displays the rooted triple $c\ell_u|a$; a contradiction. Thus, in Situations 2 and 3, we may assume $\ell_u \in \{c, d\}$, and therefore p'_{cd} lies on the path from u to p_{ab}. We can therefore resolve v_c to p_c without changing the topology of the embedding of $\mathcal{T}$. Regardless of the situation, we can delete arc (q_c, v_c), and the resulting phylogenetic network on X displays $\mathcal{T}$ if and only if $\mathcal{N}$ displays $\mathcal{T}$. A similar argument holds if p'_{cd} is of Type 1 and q'_{cd} is of Type 3, in which case, we delete (p_c, v_c). A symmetric argument holds in Situation 1 if p'_{cd} is of Type 4 and q'_{cd} is of Type 2 (or vice versa).

The remaining case is that p'_{cd} is of Type 3 and q'_{cd} is of Type 4 (or vice versa). An identical argument to that given to justify the entries in Table 2 corrsponding to deleting the leaf b in the first special case applies. In particular, this gives that $\mathcal{N}\backslash b$ displays $\mathcal{T}\backslash b$ if and only if $\mathcal{N}$ displays $\mathcal{T}$.

4. Running Time

In this section, we consider the running time of the TreeDetection algorithm. The input to the algorithm is a reticulation-visible network $\mathcal{N}$ on a finite set X and a rooted binary phylogenetic X-tree $\mathcal{T}$. Traditionally running times in phylogenetics are given in terms of the size of the taxa set, $|X|$, although the representation of a general phylogenetic network may be much larger that $O(|X|)$. However, in the case of reticulation-visible networks, the number of vertices of the network is at most $8|X| - 7$, and so the input in this case is of size $O(|X|)$ (see Theorem 1.2).

Clearly, determining if $|X| = 1$ can be done in constant time, and finding a cherry $\{a, b\}$ in $\mathcal{T}$ at maximum path length from the root can be done in linear time. The main body of the algorithm then boils down to determining which of several cases occurs, and then running a specific subroutine. We first show that we can determine which case occurs in $O(|X|^2)$ steps after preprocessing. To check the Easy Cases, we need to know the vertices v_a, v_b, and ρ_{ab}, and their ancestors and descendants. In a preprocessing step, for each vertex of $\mathcal{N}$, we prepare lists of the following:

(i) all ancestors,
(ii) all descendants,
(iii) all leaves that verify the vertex, and
(iv) all descendants reachable by a tree-path from the vertex.

The most complex and time-consuming list to construct is (iii). To find the leaves that verify a vertex v, we first construct a directed graph $\mathcal{N}'$ from $\mathcal{N}$ by deleting the arcs coming into v, then determine the leaves that are reachable from the root in $\mathcal{N}'$ via a depth-first search, and finally compute the set of leaves that are descendants of v in $\mathcal{N}$ and are not reachable from the root in $\mathcal{N}'$; this is precisely the set of leaves that verify v. This can be done, for each v, in $O(|X|^2)$ steps. We also order the vertices by distance of the vertex from the root; since there are less than $8|X|$ vertices altogether, this can be done in $O(|X|^2)$ steps using a depth-first search. We can then scan through the list in order to determine v_a, v_b, and ρ_{ab} by checking whether each vertex satisfies the conditions. Further scanning through the list and applying simple checks at each vertex can determine which, if any, of the Easy Cases occurs and, since we only need compare at most two vertices, in particular, at (EC7), this can all be achieved in $O(|X|^2)$ steps.

Depending on whether the sibling of the cherry $\{a, b\}$ is a single leaf c or another cherry $\{c, d\}$, we must then check which of several Special Cases occurs. However, once we have identified $\hat{v}_c$, or ρ_{cd}, p_{cd}, and q_{cd} by looking through the lists above, it is again a matter of checking whether vertices with certain straightforward conditions on their ancestors and descendants exist or not, and this can be done in $O(|X|^2)$ steps.

Finally, each of the subroutines either returns **No** or modifies $\mathcal{N}$ and possibly $\mathcal{T}$ before recursively calling TreeDetection. Each such modification consists of deleting a constant number of arcs and vertices. Determining whether to return **No** or make the modifications can be accomplished in $O(|X|^2)$ steps as the network is of linear size. The reason for this is that they again require at most checking the existence of vertices with specific ancestor or descendant conditions, possibly including reachability by a tree-path. This analysis includes determining which row of the final tables is appropriate, since we can identify the vertices p'_c and q'_c, or p'_{cd} and q'_{cd} by

scanning through the vertex list and checking straightforward ancestor and descendant conditions, and then determine which of the rows is applicable by again checking ancestor and descendant conditions on these vertices. For example, we can tell if a vertex is on a path from p_{ab} to p_a by checking if it is a descendant of p_{ab} and an ancestor of p_a.

Thus, after the preprocessing which takes $O(|X|^3)$ steps, the entire algorithm breaks down into a constant number of checks, each of which can be accomplished in $O(|X|^2)$ steps, followed by a small modification to $\mathcal{N}$ and possibly $\mathcal{T}$, and a recursive call to TreeDetection on an input that is smaller in either the number of reticulations or the size of the leaf set. Since the number of reticulations is linear in $|X|$, there can be at most $O(|X|)$ recursive calls, and so the entire algorithm (including preprocessing) completes in $O(|X|^3)$ steps.

5. Sharp Bounds

In this section, we establish Theorem 1.2.

Proof of Theorem 1.2. Let $\mathcal{N}$ be a reticulation-visible network on X with n vertices in total and r reticulations. Let $m = |X|$. We first show that

$$n \leq 8m - 7 \tag{1}$$

and

$$r \leq 3m - 3. \tag{2}$$

The proof of these two inequalities is by induction on m. If $m = 1$, then, by Lemma 2.1, $\mathcal{N}$ consists of a single vertex, and (1) and (2) hold. Suppose that $m \geq 2$, and that (1) and (2) hold for all reticulation-visible networks with fewer leaves.

First assume that $\mathcal{N}$ has a cherry $\{a, b\}$. Let $\mathcal{N}'$ be the network obtained from $\mathcal{N}$ by reducing $\{a, b\}$. Since every vertex in $\mathcal{N}$ is visible, it follows that every vertex in $\mathcal{N}'$ is visible, and so $\mathcal{N}'$ is a reticulation-visible network. Therefore, as $\mathcal{N}'$ has $n-2$ vertices, r reticulations, and $m-1$ leaves, it follows by the induction assumption that $n-2 \leq 8(m-1)-7$ and $r \leq 3(m-1)-3$, so

$$n \leq 8m - 13 \leq 8m - 7$$

and

$$r \leq 3m - 6 \leq 3m - 3.$$

Thus (1) and (2) hold.

Now assume that $\mathcal{N}$ does not contain a cherry. Let v be a reticulation in $\mathcal{N}$ such that amongst all reticulations in $\mathcal{N}$ it is at maximum distance

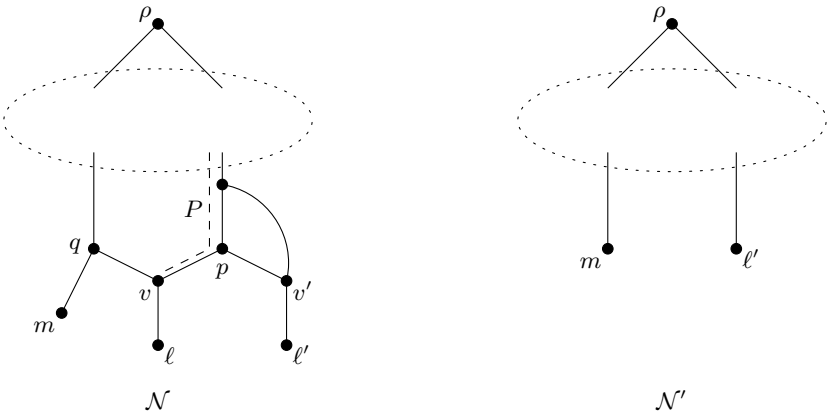


FIGURE 10. Two phylogenetic networks $\mathcal{N}$ and $\mathcal{N}'$, where $\mathcal{N}'$ has been obtained from $\mathcal{N}$ by deleting v and ℓ. As the result of creating parallel arcs after deleting v and contracting p, the phylogenetic network $\mathcal{N}'$ has two less reticulations and six less vertices in total than $\mathcal{N}$.

from the root ρ. Let P be a path from ρ to v that realises this maximum distance. By maximality and the assumption that $\mathcal{N}$ has no cherry, the child vertex of v is a leaf, ℓ say. Also, note that, as $\mathcal{N}$ is reticulation visible, neither parent of v is a reticulation. Let $\mathcal{N}'$ be the network obtained from $\mathcal{N}$ by deleting the vertices v and ℓ and their incident arcs, contracting any resulting degree-two vertices, and then replacing any parallel arcs with a single arc and contracting any degree-two vertices resulting from this replacement. See Fig. 10 for an illustration of this reduction. Since the final step in this reduction could not have created any further parallel arcs as $\mathcal{N}$ is reticulation visible, it is easily seen that $\mathcal{N}'$ is a phylogenetic network on $X - \{\ell\}$. Furthermore, in the process of obtaining $\mathcal{N}'$ from $\mathcal{N}$, we initially lose 1 reticulation and 4 vertices in total. Additionally, at most two pairs of parallel arcs are replaced with a single arc. Each such replacement, loses 1 reticulation and 2 vertices in total. Thus, if n' and r' denotes the total number of vertices and the number of reticulations in $\mathcal{N}'$, we have

$$n - 8 \leq n' \leq n - 4 \tag{3}$$

and

$$r - 3 \leq r' \leq r - 1. \tag{4}$$

We next show that either $\mathcal{N}'$ or a phylogenetic network obtained from $\mathcal{N}$ in an analogous way is reticulation visible, thereby obtaining a phylogenetic network that satisfies the induction assumptions. Let p and q denote the parents of v. If ℓ does not verify the visibility of any reticulation other than v in $\mathcal{N}$, then $\mathcal{N}'$ is reticulation visible. Therefore, suppose that ℓ also

verifies the visibility of a reticulation w in $\mathcal{N}$, where $w \neq v$. Without loss of generality, we may assume that p is the last parent of v on P. If p is an ancestor of q, then P is not a path of maximum distance from ρ to v, so either p is a descendant of q or p is non-comparable to q. In either case, this implies that p has a child vertex, not equal v, that is not an ancestor of v. Let v' denote this child vertex. By the maximality of P and the assumption that $\mathcal{N}$ has no cherries, v' is either a leaf or a reticulation. If v' is a leaf, then, as every path in $\mathcal{N}$ from ρ to ℓ passes through w, every path in $\mathcal{N}$ from ρ to v' also passes through w. It now follows that if v' is a leaf, $\mathcal{N}'$ is reticulation visible. Therefore suppose that v' is a reticulation. By the maximality of P and the assumption that $\mathcal{N}$ has no cherries, the child of v' is a leaf, ℓ' say. Let $\mathcal{N}_1'$ be the network obtained from $\mathcal{N}$ by deleting v' and ℓ' and their incident arcs, contracting any resulting degree-two vertices, and then replacing any parallel arcs with a single arc and contracting any degree-two vertex resulting from this replacement. As above, if n_1' and r_1' denote the total number of vertices and the number of reticulations in $\mathcal{N}_1'$, then

$$n - 8 \leq n_1' \leq n - 4$$

and

$$r - 3 \leq r_1' \leq r - 1.$$

If ℓ' does not verify the visibility of any reticulation other than v' in $\mathcal{N}$, then, instead of applying the inductive step to v, apply the inductive step to v'. In particular, $\mathcal{N}_1'$ is reticulation visible and therefore satisfies the induction assumptions. Thus we may assume that ℓ' also verifies the visibility of a reticulation w' in $\mathcal{N}$, where $w' \neq v'$. Now, every path in $\mathcal{N}$ from ρ to ℓ passes through w, in particular, every path in $\mathcal{N}$ from ρ to ℓ using the arc directed into p passes through w. In turn, this implies that every path in $\mathcal{N}$ from ρ to ℓ' using the arc directed into p passes through w. We deduce that either w is an ancestor of w' or w' is an ancestor of w in $\mathcal{N}$. More generally, this means that all vertices in $\mathcal{N}$ verified by either ℓ or ℓ' lie on a path Q from ρ to p. Let z be the vertex on Q verified by either ℓ or ℓ' that is at maximum distance from ρ. Without loss of generality, we may assume that z is verified by ℓ'. Then every path in $\mathcal{N}$ from ρ to ℓ' passes through each of the vertices on Q verified by either ℓ or ℓ', and so $\mathcal{N}'$ is reticulation visible, in which case, $\mathcal{N}'$ satisfies the induction assumptions.

It now follows that the induction assumptions hold for $\mathcal{N}'$. By induction, and (3) and (4), it follows that

$$n - 8 \leq n' \leq 8(m-1) - 7 = 8m - 15,$$

so $n \leq 8m - 7$, and

$$r - 3 \leq r' \leq 3(m-1) - 3 = 3m - 6,$$

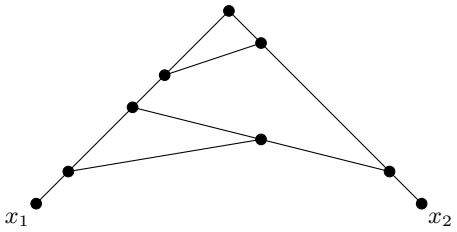


FIGURE 11. A reticulation-visible network with 2 leaves, 3 reticulations, and 9 vertices in total.

so $r \leq 3m - 3$. Hence (1) and (2) hold.

To show that the upper bounds (1) and (2) are sharp for all integers $m \geq 1$, consider the reticulation-visible network shown in Fig. 11. Here the bound is tight for when $m = 2$. For all integers $m \geq 3$, an analogous example can be constructed by replacing a leaf with a complete copy of this network. The resulting network is reticulation-visible, and the number of leaves has increased by 1, the number of reticulations has increased by 3, and the total number of vertices has increased by 8, thereby retaining the tightness of the bound. Of course, if $m = 1$, then $\mathcal{N}$ consists of a single vertex and the bounds hold exactly.

□

REFERENCES

[1] G. Cardona, F. Rossello, G. Valiente. Comparison of tree-child phylogenetic networks. *Trans. Comput. Biol. Bioinform.*, 6:552–569, 2009.

[2] P. Gambette, A. D. M. Gunawan, A. Labarre, S. Vialette, L. Zhang. Locating a tree in a phylogenetic network in quadratic time. In: *Proc. 19th Ann. Inf. Conf. Res. Comp. Mol. Biol. (RECOMB'15)*, Lecture Notes in Computer Science, Vol. 9029, pp. 96–107, 2015.

[3] A. D. M. Gunawan, B. Dasgupta, L. Zhang. Locating a Tree in a Reticulation-Visible Network in Cubic Time. To appear: *Proc. 20th Ann. Inf. Conf. Res. Comp. Mol. Biol. (RECOMB'16)*, 2016.

[4] L. van Iersel, C. Semple, M. Steel. Locating a tree in a phylogenetic network. *Inform. Process. Lett.*, 110:1037–1043, 2010.

[5] I. Kanj, L. Nakhleh, C. Than, G. Xia. Seeing the trees and their branches in the network is hard. *J. Theor. Comp. Sci.*, 401:153–164, 2008.

[6] C. Semple, M. Steel, Phylogenetics, Oxford University Press, 2003.

School of Engineering Computer Sciences, Durham University, Durham DH1 3LE, United Kingdom

E-mail address: m.j.r.bordewich@durham.ac.uk

Biomathematics Research Centre, School of Mathematics and Statistics, University of Canterbury, Christchurch, New Zealand

E-mail address: charles.semple@canterbury.ac.nz